# Deep Learning for Chaos Detection

Roberto Barrio,[1] Álvaro Lozano,[2] Ana Mayora-Cebollero,[1] Carmen Mayora-Cebollero,[1] Antonio Miguel,[3] Alfonso Ortega,[3] Sergio Serrano,[1] and Rubén Vigara[1]

[1)] *Departamento de Matemática Aplicada and IUMA. Computational Dynamics group. Universidad de Zaragoza. E-50009. Spain.*

[2)] *Departamento de Matemáticas and IUMA. Computational Dynamics group. Universidad de Zaragoza. E-50009. Spain.*

[3)] *Departamento de Ingeniería Electrónica y Comunicaciones. Voice Input Voice Output Laboratory. Universidad de Zaragoza. E-50018. Spain.*

(*Electronic mail: rvigara@unizar.es)

(*Electronic mail: sserrano@unizar.es)

(*Electronic mail: ortega@unizar.es)

(*Electronic mail: amiguel@unizar.es)

(*Electronic mail: cmayora@unizar.es)

(*Electronic mail: amayora@unizar.es)

(*Electronic mail: alozano@unizar.es)

(*Electronic mail: rbarrio@unizar.es)

(Dated: June 30, 2023)

In this article we study how a chaos detection problem can be solved using Deep Learning techniques. We consider two classical test examples: the Logistic map as a discrete dynamical system, and the Lorenz system as a continuous dynamical system. We train three types of Artificial Neural Networks (Multi-Layer Perceptron, Convolutional Neural Network and Long Short-Term Memory cell) to classify time series from the mentioned systems into *regular* or *chaotic*. This approach allows us to study biparametric and triparametric regions in the Lorenz system due to their low computational cost compared to traditional techniques.

*Keywords*: chaos detection, Deep Learning, Lyapunov Exponents, Logistic map, Lorenz system

*AMS codes*: 34D08, 37M10, 68T07.

**Deep Learning techniques have recently been introduced in the area of Dynamical Systems. These new tools can speed up studies and permit us to go deeper into simulations. Of all the problems in which these methodologies can help us, we focus on the problem of detecting chaos, showing how Deep Learning allows, in a fast way, to handle large amounts of data, such as 2D and 3D parametric phase space studies, and therefore they can be powerful techniques in global analysis.**

# I. INTRODUCTION

One of the main topics in Dynamical Systems is the detection of chaotic regions in the parameter space. The classical technique to detect chaos is the use of the Lyapunov Exponents (LEs)[1,2,4]. Recently, some authors have applied Deep Learning (DL) techniques in Dynamical Systems to handle different tasks, mainly for forecasting problems[7,19], but also for chaos detection[3,6,14]. Here, we are interested in the latter of these tasks.

In this paper, we choose three common DL architectures (Multi-Layer Perceptron, Convolutional Neural Network and Long Short-Term Memory cell) for chaos detection in time series from a dynamical system. We provide a detailed analysis of the learning process of the networks, and we are able to use the trained networks to reproduce 1D, 2D and 3D parametric plots. Remarkably, we are able to obtain the behaviour of a dynamical system in regions of the parameter space where the DL techniques have not been trained. Moreover, as far as we know, this is the first time in the literature that a dense 3D parametric plot of a continuous dynamical system, such as the Lorenz system, is represented.

This paper is organized as follows. In Section II we describe the chosen DL networks giving a brief introduction of each of them. In Sections III and IV we perform the chaos detection task in the Logistic map and the Lorenz system, respectively. We give a detailed description of the datasets and network architectures, and comment on the obtained results. In Subsection IV E we present 2D and 3D parametric diagrams of the Lorenz system computed with the trained networks. Finally, we draw some conclusions in Section V.

All the DL experiments in this paper have been performed using PyTorch[17]. The code was executed on a Linux box with dual Xeon ES2697 with 128Gb of DDR4-2133 memory with a RTX2080Ti GPU.

## II. INTRODUCTION TO THE DEEP LEARNING ARCHITECTURES USED TO DETECT CHAOS

Deep Learning[9,11] is the branch of Machine Learning that uses Deep Artificial Neural Networks to learn from data with several levels of abstraction. Artificial Neural Networks (ANNs) are formed by artificial neurons (loosely inspired by their biological counterparts) organized in layers.

Of all the DL architectures found in the literature, the Multi-Layer Perceptron (MLP) is the simplest one and it is widely used for this reason. Convolutional Neural Networks (CNNs) and Long Short-Term Memory networks (LSTMs) have been previously used to analyse time series data[3,6,14], as in our chaos detection experiments. We have tested these three well-known ANN architectures to detect chaos.

For the MLP, we start with a similar architecture to that used in Ref. 3. For the CNN and LSTM, we use a not very complicated structure and we do not perform hyperparameter optimization. A more detailed study of the network architectures may improve our results and it is part of our future research.

**Remark II.1.** *Although our mathematical problem is called* chaos detection*, from the point of view of DL it is a* binary classification task *instead of a detection task: our networks classify the input vectors (time series corresponding to an orbit of a dynamical system) into two disjoint categories (*regular *vs* chaotic*).*

*Multi-Layer Perceptron.* One of the fundamental Deep Learning architectures is the Multi-Layer Perceptron[9]. It operates by taking a linear combination of the inputs in each layer, followed by a non-linear activation function. In Figure 1A we have an example of an MLP whose output $y$ is given by

$$y = W^{[3]}\mathscr{A}(W^{[2]}\mathscr{A}(W^{[1]}x + b^{[1]}) + b^{[2]}) + b^{[3]},$$

where $x$ is the input, $W^{[l]}$ is the matrix of weights of the connections between layer $l-1$ and layer $l$ (layers are enumerated from 0 to 3 from left to right), $b^{[l]}$ represents the bias vector of layer $l$, and $\mathscr{A}$ is a non-linear activation function such as the Rectified Linear Unit $\mathrm{ReLU}(x) = \max(0,x)$, the sigmoid $\sigma(x) = (1+e^{-x})^{-1}$ or the hyperbolic tangent $\tanh(x) = 2\sigma(2x) - 1$.

*Convolutional Neural Network.* Convolutional Neural Networks were originally developed for image recognition tasks[13], and are organized into convolutional and pooling layers to capture features and reduce dimensions, respectively. One of the key features of CNNs is that they share

Networks.pdf

Figure 1. Simple graphic representations of the architectures of (A) an MLP with two hidden layers, (B) a 1D CNN with three channels in the represented convolutional layers, and (C) a generic LSTM cell.

weights across multiple neurons[5] for more efficient processing. They handle different input formats such as vectors, matrices, or 3D tensors, depending on the type of convolution used. In this paper, the input data is in vector form, and therefore we focus on the use of 1D CNNs, as depicted in Figure 1B.

To exemplify how a CNN works, we show how to compute the value of the shaded neuron in the second layer of the network in Figure 1B, which is given by

$$x_{0,0}^{[1]} = \mathscr{A}\left(b_0^{[1]} + \sum_{j=0}^{1}\sum_{k=0}^{2} w_{j,k,0}^{[1]} x_{j,k}^{[0]}\right),$$

where $x_{j,k}^{[l]}$ is the activation of neuron $j$ of channel $k$ at layer $l$ (the first index for the neurons, the channels and the layers is 0), $W_0^{[1]} = (w_{j,k,0}^{[1]})_{j=0,1;k=0,2}$ is the weight matrix, $b_0^{[1]}$ is the bias vector, and $\mathscr{A}$ is the activation function. We could have more complex CNN architectures[10,18] if we consider stride, dilation, residual connections, …

*Long Short-Term Memory.* Recurrent Neural Networks (RNNs) are commonly used for sequential processing since they retain some information from past inputs. Long Short-Term Memory cells[12] represent a specific type of RNN architecture. Among the distinctive elements of an LSTM are the hidden state $h$ and the cell state $c$, which are the elements keeping information from previous steps. Computations performed by such type of memory cells (represented in Figure 1C) are

$$\begin{aligned}
f(t) &= \sigma\left(W_f^{[x]}x(t) + W_f^{[h]}h(t-1) + b_f\right), & g(t) &= \tanh\left(W_g^{[x]}x(t) + W_g^{[h]}h(t-1) + b_g\right),\\
i(t) &= \sigma\left(W_i^{[x]}x(t) + W_i^{[h]}h(t-1) + b_i\right), & c(t) &= f(t)\otimes c(t-1) + i(t)\otimes g(t),\\
o(t) &= \sigma\left(W_o^{[x]}x(t) + W_o^{[h]}h(t-1) + b_o\right), & y(t) &= h(t) = o(t)\otimes \tanh(c(t)),
\end{aligned} \tag{1}$$

where $x(t)$ is the external input, $y(t)$ is the usual output, $W_*^{[x,h]}$ and $b_*$ represent the matrix of weights and the bias term for the external input or the hidden state, $\otimes$ is the element-wise product, and $\sigma$ and tanh are the activation functions. Roughly speaking, as a consequence of the application of the sigmoid activation function in left formulas in (1), $f$, $i$ and $o$ are deciding which information is kept and which is removed in the output and the hidden and cell states.

## III. TEST EXAMPLE OF DISCRETE DYNAMICAL SYSTEMS: THE LOGISTIC MAP

The Logistic map[16] is a well-studied model that describes the dynamics of animal populations. It is given by the equation

$$x_{n+1} = \alpha x_n(1-x_n), \tag{2}$$

where $x_n$ is the variable in the $n$-th iteration and $\alpha$ is the bifurcation parameter. Note that since (2) is symmetric with respect to $x = 0.5$, the evolution of the initial condition $x_0$ is exactly the same as the one of $1-x_0$. The Lyapunov Exponent for this map is

$$\mathrm{LE} = \lim_{k\to+\infty} \frac{1}{k}\sum_{n=0}^{k-1} \log|\alpha(1-2x_n)|, \tag{3}$$

where log is the natural logarithm.

### A. Dataset

To prepare the training, test, and validation sets for the Logistic map, we generate multiple raw datasets of time series samples, which are subsequently screened. Each raw dataset comprises time series with a fixed length of 1000. The initial condition $x_0$ is fixed for all the time series in each raw dataset ($x_0 \in \{0.5, 0.9\}$ for training dataset, $x_0 = 0.75$ for validation, and $x_0 = 0.8$ for test), and the bifurcation parameter $\alpha$ takes 12000 equidistant values in the interval $[0,4)$. The time series of the raw datasets are the last 1000 time steps obtained applying the iterative formula (2) for 12000 time steps. The LE of each sample is approximated applying formula (3) with the last 11000 time steps of the time series (the first 1000 time steps are considered as transient time).

From the union of the raw datasets with $x_0 = 0.5$ and $x_0 = 0.9$, we obtain the training dataset. In particular, we split the joint dataset into regular and chaotic samples (chaotic if the approximated LE is greater than 0.1, to reduce the transient behaviour as done in several studies; and regular otherwise), obtaining 21791 regular and 2209 chaotic time series. We delete the samples that are similar (we consider that two samples are similar if their distance in infinity norm is less than $10^{-4}$) to avoid repeated time series. In this process, the number of chaotic samples is not reduced, but the number of regular samples decreases to 6402. After shuffling these datasets (to have time series from different space regions in the subsequent selection), we choose 2000 chaotic and 2000 regular samples for the training set.

Validation dataset is obtained from raw dataset with $x_0 = 0.75$ (it contains 10896 regular and 1104 chaotic samples). The training, test and validation datasets have to be pairwise disjoint. So, we drop any validation sample similar in the previous sense to a training sample. After this process, we have 1268 regular and 1104 chaotic time series. We build the validation dataset with 1000 regular and 1000 chaotic samples randomly taken.

For test dataset we use raw dataset with $x_0 = 0.8$ (it has 10896 regular and 1104 chaotic time series). After data selection, the number of chaotic samples does not change, but the number of regular ones decreases to 5705. After shuffling, we build the test dataset with 1000 regular and 1000 chaotic samples.

To perform a supervised DL training as the one that concerns us, we need the network inputs (that we have already created) and the expected labels, i.e., whether the inputs are regular or chaotic. We label a times series with a 0 if it is regular and with 1 if it is chaotic. To input the data into the networks, we split each dataset into different batches. The batch sizes of the training, validation and test sets are 128, 100 and 100, respectively (in the case of the training set, the last incomplete batch is deleted).

## B. Multi-Layer Perceptron

Our architecture is inspired by the one in Ref. 3, with some changes as the overfitting technique (instead of dropout, we perform early stopping and $L^2$-regularization). It has 5 layers: an input layer with 1000 neurons (the length of the input), three hidden layers with 500 neurons each one, and an output layer with 2 neurons (regular vs chaotic). The ReLU activation function is applied in the hidden layers, and the softmax function in the output layer. This network is trained for 2000 epochs, saving the first model with the lowest value of the loss function for the validation dataset (early stopping technique). Here we use the Cross-Entropy Loss with weight decay $10^{-5}$ for the $L^2$-penalty, optimized with the Stochastic Gradient Descent with learning rate $10^{-3}$.

## C. Convolutional Neural Network

Our CNN architecture has two 1D convolutional layers with 5 and 10 channels, kernel size 10 and 5, dilation equal to 2 and 4, respectively; and stride 1 for both. Each convolutional layer adds a bias term and applies the ReLU activation function. We use zero-padding and cropping to ensure that the length of the output sequence remains the same as the input since the stride is equal to one. A global average pooling layer is applied after the last convolutional layer. A binary classification layer with 2 neurons is stacked at the end. A bias term is added and the softmax activation function is applied. The CNN is trained for 2000 epochs using the early stopping technique. The loss function is the Cross-Entropy Loss with weight decay $10^{-5}$ for the $L^2$-regularization. For this architecture, the optimizer is Adam with learning rate 0.008.

## D. Long Short-Term Memory

Our LSTM architecture consists of 2 stacked layers followed by a linear layer with two output neurons for classification. Both LSTM layers are unidirectional, their states have dimension 4 and bias terms are considered. The input of the linear layer is the last hidden state of both LSTM cells. The network is trained for 2000 epochs with early stopping. The loss function and the optimizer are the same as the ones used for the CNN in Subsection III C.

## E. Results

The accuracy is the measure used to determine the performance of all the networks. It is computed with the following formula

$$\text{Accuracy (\%)} = \frac{T_R + T_C}{T_R + T_C + F_R + F_C} \cdot 100,$$

where $T_R$ and $T_C$ represent the number of regular and chaotic samples, respectively, that the network has classified correctly; $F_R$ and $F_C$ are the number of chaotic and regular samples, respectively, that have been wrongly classified. To assure that the network has learnt correctly, that is, the percentage of correct classifications of both classes is balanced, the following magnitudes are computed

$$\text{Accuracy Regular (\%)} = \frac{T_R}{T_R + F_C} \cdot 100 \quad \text{and} \quad \text{Accuracy Chaotic (\%)} = \frac{T_C}{T_C + F_R} \cdot 100.$$

**Remark III.1.** *The performance results of the networks indicated in this subsection and Subsections IV C-IV D are the mean and the standard deviation of the binary classification of* 10 *trained networks randomly initialized by PyTorch (the graphic results correspond to a unique network). The experiments in Subsection IV E are carried out using one of the trained networks.*

| | MLP | | CNN | | LSTM | |
|---|---|---|---|---|---|---|
| | **Loss** | **Accuracy (%)** | **Loss** | **Accuracy (%)** | **Loss** | **Accuracy (%)** |
| **Training** | $0.026 \pm 0.001$ | $99.302 \pm 0.161$ | $0.018 \pm 0.007$ | $99.383 \pm 0.335$ | $0.059 \pm 0.020$ | $97.979 \pm 0.584$ |
| **Validation** | $0.107 \pm 0.003$ | $96.775 \pm 0.068$ | $0.037 \pm 0.004$ | $98.925 \pm 0.203$ | $0.104 \pm 0.048$ | $96.605 \pm 1.337$ |
| **Test** | $0.046 \pm 0.002$ | $97.865 \pm 0.098$ | $0.018 \pm 0.004$ | $99.410 \pm 0.239$ | $0.045 \pm 0.013$ | $98.565 \pm 0.279$ |

Table I. Loss and accuracy (%) of training, validation and test datasets for the Logistic map test problem. Results in this and similar tables or figures are given as mean±standard deviation for 10 different trials (see Remark III.1).

We show in Table I the loss and accuracy of training, validation and test datasets for the three ANNs (MLP, CNN and LSTM). Notice that for all the networks, the training and test losses are quite small, so we can consider that the ANNs have been able to learn from data correctly. If we compare the results for the test data, we can see that our CNN seems to give better results than the other two networks. For completion, we can compute the accuracy in the regular and chaotic samples of test set, obtaining the results in Table II. As we can see, the percentages of regular and chaotic samples are quite similar (and always close to 100%), so our ANNs have learnt correctly the main features of both types of dynamical regimes.

| | **MLP** | **CNN** | **LSTM** |
|---|---|---|---|
| **Regular** | $99.860 \pm 0.049$ | $99.710 \pm 0.094$ | $99.030 \pm 0.827$ |
| **Chaotic** | $95.870 \pm 0.200$ | $99.110 \pm 0.509$ | $98.100 \pm 0.775$ |

Table II. Accuracy (%) of regular and chaotic samples in the test set for the Logistic map test problem.

We can also analyse the evolution of the loss and accuracy of the training and validation datasets along the 2000 epochs to study the learning process of the networks. To exemplify it, we choose one of the trained MLPs (for all the trained networks, a similar situation occurs). In Figure 2 we have drawn the evolution of the loss and accuracy of training (blue) and validation (red) datasets along 2000 epochs. Notice that from epoch 1000 (approximately), the loss and the accuracy of the training dataset do not change significantly, so although we can see that the mathematical optimum of the loss function for the validation dataset has been obtained at epoch 1770 (best epoch, see green line in Figure 2), we could say that the computational optimum has been achieved in epoch 1000 approximately (see purple line in Figure 2). We can conclude that we could train the network for less number of epochs and the obtained minimum of the loss function would be similar and still acceptable (moreover, the training time would be less).

loss_acc_MLP_lm.pdf

Figure 2. Evolution along 2000 epochs of the loss (left panel) and the accuracy (right panel) in training (blue) and validation (red) datasets for one MLP trained in the Logistic map test problem. Green line corresponds to the best epoch (1770) and purple line to the recommended epoch (1000).

Once we have successfully trained our networks, we carry out the chaos detection task in one parametric line of the Logistic map: we fix $x_0 = 0.6$ (we cannot take $x_0 \in \{0.1, 0.2, 0.25, 0.5, 0.75, 0.8, 0.9\}$ as these are the values used to create the datasets and their symmetric cases with respect to $x = 0.5$), we take 12000 values of $\alpha \in [0, 4)$, and we use each network to detect the behaviour of the system. In Figure 3 we have a table with the accuracy of the networks for each dynamical regime. We have also drawn the approximated LEs in red (that have been computed with a classical technique as explained in Subsection III A) and in the three colour bars we have the results given by the networks. Light colours are used for regular regimes, and dark ones, for chaotic behaviour. Blue is the colour chosen for the MLP, pink for the CNN, and green for the LSTM. As we can see in the aforementioned table, all the networks are able to perform the chaos detection task: the accuracy is close to 100% for all the architectures and dynamical regimes. Furthermore, in the three colour bars we can see that all the networks are able to detect quite well the boundaries between both dynamical regimes (see the first dotted line from left to right), and they have detected the regular windows in the chaotic region (see the remaining dotted lines in the image).

In Figure 4 we have a histogram with the percentage of correctly classified time series in the $\alpha$-parametric line according to the value of the approximated LE (100% means that all the samples whose LE is in the corresponding interval have been correctly detected). Blue bars correspond to the MLP, pink ones to the CNN, and green ones to the LSTM network. Notice that when the LE is far from 0, the three networks perform the task perfectly. Otherwise, the percentage of correct detections is lower but still larger than 90%, with the MLP giving the worst results.

It is interesting to analyse some correct and incorrect classifications of the DL networks (see yellow points in Figure 3). In case I of Figure 5 we show a periodic orbit (regular behaviour)

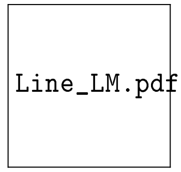


Figure 3. Chaos detection task with the MLP, the CNN and the LSTM in an $\alpha$-parametric line ($x_0 = 0.6$) of the Logistic map test problem. From top to bottom, three colour bars with the results (light for regular, dark for chaotic) of the trained networks (MLP in blue, CNN in pink and LSTM in green), approximated LEs, and table with the accuracy of regular and chaotic samples for the three types of architectures. In yellow, $\alpha$ values of the orbits in Figure 5.

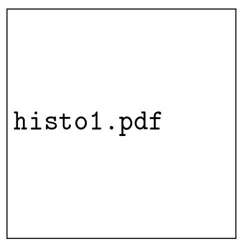


Figure 4. Histogram of the percentage of correct detections according to the value of the approximated LE (blue for the MLP, pink for the CNN, and green for the LSTM) for the $\alpha$-parametric line with $x_0 = 0.6$ (see Figure 3) of the Logistic map test problem.

time series that all the networks have been able to classify correctly. The sample of case II is also regular, but in this case, it is misclassified by all the models. In case III, the orbit is chaotic and only the CNN detects it correctly. Finally, in case IV, the sample is chaotic too and all the networks classify it correctly. Case II illustrates a periodic orbit with many oscillations, being extremely difficult to recognise as it has a similar behaviour to some chaotic samples in the dataset. Case III illustrates another common dynamical situation, a chaotic behaviour but where the chaos is weak, that is, the "irregularity" is small, being quite similar to a periodic orbit, and so, the DL techniques can also be wrong. Note that these two cases with wrong detections are also complicated cases for standard techniques. In these cases expert researchers would use their background to classify the behaviour. In any case, these boundary cases are just a small percentage of the tests, most of the data is correctly classified as regular or chaotic.

Orbitas1a.pdf

Figure 5. Some orbits of the Logistic map, its true dynamical behaviour and the success or error of the classification made by the different networks. For a correct visualization, only the first 200 steps of the Logistic map have been drawn. In all cases $x_0 = 0.6$, and (I) $\alpha = 3.43233323097229$, (II) $\alpha = 3.6559998989105225$, (III) $\alpha = 3.856666549414062$ and (IV) $\alpha = 3.999666690826416$ (see yellow points in Figure 3).

## IV. TEST EXAMPLE OF CONTINUOUS DYNAMICAL SYSTEMS: THE LORENZ SYSTEM

The Lorenz system[15] is a very simple model representing cellular convection. It is given by the system of equations

$$\begin{cases} \dot{x} &= \sigma(y-x), \\ \dot{y} &= -xz+rx-y, \\ \dot{z} &= xy-bz, \end{cases}$$

where $x$, $y$ and $z$ are the variables, $\sigma$ is the Prandtl number, $r$ is the relative Rayleigh number, and $b$ is a positive constant. The Lyapunov Exponents of a continuous dynamical system are computed with the algorithm in Ref. 20.

### A. Dataset

As in the case of the Logistic map, we create several raw datasets (with time series of the Lorenz system), and then we screen them to obtain the three sets needed to train, validate and test the networks. Each raw dataset satisfies that the length of the time series is 1000 (in Subsection IV D we justify our choice), and the initial condition is fixed to $(x_0, y_0, z_0) = (1, 1, 1)$ (the initial value used by other authors[3] in this chaos detection task). Bifurcation parameter $\sigma$ is fixed to the classical value 10, the relative Rayleigh number $r$ moves equidistantly in the interval $(0, 300]$ to obtain 5999 values, and the positive constant $b$ is fixed along all the samples of each raw dataset ($b \in \{2, 8/3\}$ to build the training dataset, $b = 2.4$ for validation, and $b = 2.8$ for test). A transient interval is performed until time $t = 100000$ with time step 0.01 using DOPRI5 (Runge-Kutta integrator of order 5); the LEs are computed using 10001 more unit times with time step 0.001 and the same integrator; the time series that we use as input in the DL architectures are built with 1 out of every 100 of the last 100000 computed points.

From the union of raw datasets with $b = 2$ and $b = 8/3$ we obtain the training set. In particular, we split the joint dataset into regular and chaotic samples taking 0.01 as threshold (chaotic if the first approximated LE is bigger than 0.01, and regular otherwise), obtaining 4091 regular and 7907 chaotic time series. To ensure variability, we delete all but one similar samples (that is, samples at distance less than $10^{-4}$ in the infinity norm). From the resulting set with 4054 regular and 7907 chaotic samples, we randomly choose 3900 samples of each dynamical class as training dataset. To build the validation dataset, we consider the raw dataset with $b = 2.4$, with 2278 regular and 3721 chaotic samples. From it, we remove all samples similar (in the previous sense) to any of the training set (2259 regular and 3721 chaotic samples remain) and we choose 1000 of each class as validation dataset. Finally, from the 2902 regular and 3097 chaotic samples forming the raw dataset with $b = 2.8$, we remove all similar samples to any from the previous sets. From the resulting set (with 2884 regular and 3097 chaotic samples) we select 1000 from each class to create the test set.

As with the Logistic map, regular samples are labelled with 0, and chaotic ones with 1. The batch sizes of the training, validation and test sets are 128, 100 and 100, respectively (the last incomplete batch of the training set is deleted). In the case of the Lorenz system, we normalize each coordinate independently, linearly mapping its range to the interval $[0,1]$. If a coordinate is constant, we assign to it a random number uniformly sampled from $[0,1]$.

### B. Multi-Layer Perceptron, Convolutional Neural Network and Long Short-Term Memory

The only change in the architecture of the MLP that we have considered for the Lorenz system case with respect to the one used for the Logistic map is the number of neurons in each layer (except in the output one): 3000 neurons in the input layer (take into account that we still consider the length of each sample equal to 1000, but Lorenz system has dimension 3, so the flattened time series has length 3000) and 1500 on each hidden layer.

As in the case of the MLP, the CNN used for the Lorenz system is similar to the one used for the Logistic map. The number of channels in the convolutional layers has changed: 15 for the first one and 30 for the second. Moreover, the input layer has now 3 channels instead of 1 (one for each

| | MLP | | CNN | | LSTM | |
|---|---|---|---|---|---|---|
| | **Loss** | **Accuracy (%)** | **Loss** | **Accuracy (%)** | **Loss** | **Accuracy (%)** |
| **Training** | $0.072 \pm 0.002$ | $98.844 \pm 0.074$ | $0.046 \pm 0.011$ | $98.194 \pm 0.464$ | $0.045 \pm 0.006$ | $98.125 \pm 0.271$ |
| **Validation** | $0.194 \pm 0.002$ | $94.125 \pm 0.189$ | $0.063 \pm 0.006$ | $97.720 \pm 0.205$ | $0.042 \pm 0.005$ | $98.120 \pm 0.200$ |
| **Test** | $0.179 \pm 0.002$ | $93.350 \pm 0.613$ | $0.085 \pm 0.071$ | $97.575 \pm 0.990$ | $0.051 \pm 0.030$ | $97.870 \pm 1.326$ |

Table III. Loss and accuracy (%) of training, validation and test datasets for the Lorenz system test problem.

variable of the system).

In the case of the LSTM, we use the same Deep Learning architecture described for the Logistic map, but now the dimension of the states is 24. Again, we have to take into account the structure of the input because now we work with a three dimensional dynamical system, so the external input of the LSTM is of size 3 instead of 1 (at each time step).

### C. Results

In Table III we show the loss and accuracy of training, validation and test datasets for the MLP, the CNN and the LSTM. For all the networks, the training and test losses are quite small, so we can consider that they have been able to learn from the data correctly. If we compare the results for the test data, we can see that the CNN and the LSTM seem to give better results than the MLP. If we compare the CNN and the LSTM, they show similar performance results. For completion, the accuracy of the regular and chaotic samples of the test set are shown in Table IV. Notice that the percentages of both classes are similar for all the trained ANNs, so we conclude that the networks have learnt correctly the main features of both dynamical regimes.

| | **MLP** | **CNN** | **LSTM** |
|---|---|---|---|
| **Regular** | $92.680 \pm 0.481$ | $97.710 \pm 1.724$ | $97.770 \pm 2.759$ |
| **Chaotic** | $94.020 \pm 0.306$ | $97.440 \pm 0.709$ | $97.970 \pm 0.438$ |

Table IV. Accuracy (%) of regular and chaotic samples in the test set for the Lorenz system test problem.

Now, we analyse the evolution of the loss and the accuracy of training and validation datasets along 2000 epochs to study how this learning process is. In the case of the MLP and the CNN, the networks are able to learn with a small number of epochs as the best epoch is usually less than 750. If we visualize such evolution (see Figure 6 for an example of a trained MLP) it seems that the network has learnt as much as it can because, after the best epoch, the validation loss increases while the training loss decreases (the network suffers overfitting). In the case of the LSTM (see Figure 7 for an example of a trained LSTM network), the evolution is more similar to that of the Logistic map since acceptable results would be obtained with fewer number of epochs (the minimum of the validation dataset would be similar and less computational time would be needed). In the figure, the best epoch is 1848, but around epoch 250 the loss has already reached a value close to zero and the accuracy is close to 100% (so, 255 is a recommended epoch taking into account the little improvement of going up to 1848). Some erratic jumps highlight in this learning process. This phenomenon can be explained by the fact that some samples (see for example Figure 14I) have a difficult dynamical behaviour (they are in the limit between the regular and chaotic dynamics), which causes the networks to have trouble in learning them.

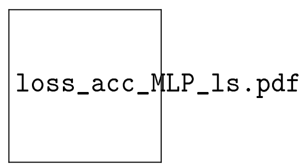


Figure 6. Evolution along 2000 epochs of the loss (left panel) and the accuracy (right panel) in training (blue) and validation (red) datasets for one MLP trained in the Lorenz system test problem. Green line corresponds to the best epoch (337).

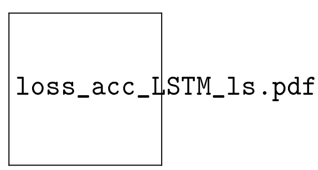


Figure 7. Evolution along 2000 epochs of the loss (left panel) and the accuracy (right panel) of the training (blue) and validation (red) datasets for one LSTM trained in the Lorenz system test problem. Green line corresponds to the best epoch (1848) and purple one to recommended epoch (255).

All the networks have learnt correctly from the data. To show the performance of the three DL architectures, we choose an $r$-parametric line of this continuous dynamical system different from the ones used to create train, test and validation sets. In particular, we take $\sigma = 10$, $b = 2.2$ and 6000 equidistant values of $r$ in the interval $[0, 300]$. In Figure 8 we have a table with the accuracy achieved by each architecture for each dynamical regime (regular and chaotic). Moreover, the first approximated LE is represented in red in the middle of the figure. In the three colour bars the results given by each network (same code of colours as in Figure 3) are depicted. It can be seen that all the networks are able to perform the chaos detection task in the Lorenz system test problem (the accuracy is greater than 90% in all cases). In addition, in the colour bars we see that all the ANNs can define correctly the boundary between both dynamical regimes in most cases (see first, seventh and eighth dotted lines from left to right). However, the networks are not able to detect the last boundary (represented with the last dotted line) perfectly (the LSTM gives the best detection), but the results are quite acceptable. Notice that the MLP shows, in general, noisier results (see region before first dotted line, between seventh and eighth dotted lines, and around last dotted line in the blue bar). All the DL architectures can detect most of the regular windows in the chaotic regions (see remaining dotted lines).

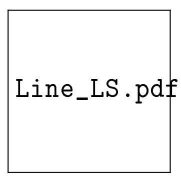


Figure 8. Chaos detection task performed by the MLP, the CNN and the LSTM in the Lorenz system test problem in an $r$-parametric line. From top to bottom, three colour bars representing the results (light for regular, dark for chaotic) of the trained networks (MLP in blue, CNN in pink and LSTM in green), first approximated LE, and table with the accuracy of regular and chaotic samples for the three types of architectures.

In Figure 9 we have a histogram that represents the percentage of correctly detected time series in the $r$-parametric line according to the value of the LE (100% means that all the samples whose first approximated LE is in the corresponding interval have been correctly classified). Blue, pink and green bars correspond to the MLP, the CNN and the LSTM networks respectively. Note that when the first LE is far from 0, the CNN and the LSTM networks perform the task perfectly, and the MLP fails for negative values. Otherwise, the percentage of correct detections is lower for all the networks but still larger than 82% in all the cases. Overall, the MLP shows the worst results, and the LSTM gives the best ones in general.

histo2.pdf

Figure 9. Histogram of the percentage of correct detections according to the value of the approximated LE (blue for the MLP, pink for the CNN, and green for the LSTM) for the $r$-parametric line with $\sigma = 10$, $b = 2.2$ (see Figure 8) of the Lorenz system test problem.

### D. Going Deeper into the Lorenz Dataset. Changing the Length

To train the networks for chaos detection in the Logistic map and the Lorenz system, we have fixed the length of the input time series to 1000 (the same as other authors have considered for this task[3]). The goal of this subsection is to show that this length seems to be a good choice for this problem. For example, in the case of the MLP, the length of the input determines the number of neurons in the input layer, so if we change it, we have to modify the architecture that we have set up in PyTorch. For the LSTM, the length can be variable, that is, we can train the same LSTM architecture with different input lengths. For this reason, we consider that it is interesting to compare the performance of our LSTM when it is trained with time series of different lengths (it tells us how much information is needed by the built LSTM network to learn the task).

ts_image2_length.pdf

Figure 10. Orbits I (panel A) and V (panel B) of Figure 14 for time series length 500 (left), 1000 (middle), and 1500 (right) of the Lorenz system test problem.

We perform the experiment (with the LSTM architecture) for length 500 and 1500, comparing with the current results for length 1000. To obtain the new datasets we have carried out the same integration steps as in the case of length 1000, computing the same number of points, but storing the last 500 or 1500. As an example, in Figure 10 orbits I and V of Figure 14 are represented with the aforementioned lengths.

In Table V we have the results for the three different lengths (information of length 1000 corre-

| | Length 500 | | Length 1000 | | Length 1500 | |
|---|---|---|---|---|---|---|
| | **Loss** | **Accuracy (%)** | **Loss** | **Accuracy (%)** | **Loss** | **Accuracy (%)** |
| **Training** | $0.050 \pm 0.005$ | $97.858 \pm 0.239$ | $0.045 \pm 0.006$ | $98.125 \pm 0.271$ | $0.057 \pm 0.037$ | $97.703 \pm 1.279$ |
| **Validation** | $0.049 \pm 0.004$ | $97.980 \pm 0.222$ | $0.042 \pm 0.005$ | $98.120 \pm 0.200$ | $0.054 \pm 0.028$ | $97.825 \pm 0.972$ |
| **Test** | $0.054 \pm 0.012$ | $97.855 \pm 0.652$ | $0.051 \pm 0.030$ | $97.870 \pm 1.326$ | $0.062 \pm 0.031$ | $97.315 \pm 1.711$ |

Table V. Loss and accuracy (%) of training, validation and test datasets with different time series length for the Lorenz system test problem.

sponds to Table III). Even though the change in the mean of the loss and the mean of the accuracy is not very considerable, length 1000 gives us the lower mean for the loss and the greatest mean for the accuracy in all the datasets. If we focus on the standard deviation, we can see that the case of length 1500 is always the one with the highest values. Cases of length 500 and 1000 present a similar standard deviation in the training and validation datasets. One possible answer for not getting an improvement for length 1500 over length 1000 is the restricted memory capacity of the LSTM recursive structures, which could be affected by long observation sequences and result in the loss of initial evidence. Note that dynamically both datasets provide similar information as it can be seen in Figure 10, unlike the case of using length 500 where the information is not complete. Therefore, we conclude that length 1000 is a good choice to perform the chaos detection task in our problem.

### E. 2D and 3D Parametric Diagrams

In a dynamical study of a mathematical model it is very interesting to perform 2D and 3D parametric analyses. In the case of the Lorenz system, several 2D studies have been presented in the literature[1,2]. Therefore, a challenging problem is to see the behaviour of the DL networks for 2D, and even 3D, parametric studies.

Just three $r$-parametric lines on the biparametric plane $(r,b)$ have been involved in the training process of DL networks (see orange and purple lines in panel A of Figure 11). As shown in Section IV C, these trained networks have performed correctly the chaos detection task in a different $r$-parametric line of the aforementioned plane. Now, we show that they are able to generalize to the whole biparametric plane $(r,b)$ with accurate results. In particular, biparametric plane $(r,b)$ with $\sigma = 10$, $r \in [0,300]$ and $b \in [2,3]$ is studied. It is important to remark that with the data of few lines, DL networks are able to analyse correctly complete biparametric planes.

p1_def_nueva.pdf

Figure 11. Chaos detection for the Lorenz system in the biparametric plane $(r,b)$ with $\sigma = 10$. To obtain the plot of panel A the first approximated LE is used to determine the regular (black) and chaotic (white) regions. The $r$-parametric lines used to create train ($b \in \{2, 8/3\}$) and validation dataset ($b = 2.4$) are shown in orange and purple, respectively. Panels B, C and D show the results of the MLP, the CNN and the LSTM network, respectively. On the left, classification of the networks (again, regular in black and chaotic in white); on the right, errors committed by the architectures (red for false regulars, blue for false chaotics). The percentage corresponds to the accuracy of the corresponding DL technique. In panels A1, B1, C1 and D1, a zoom of the green framed region in the biparametric plane has been represented for each method. The yellow points correspond to orbits in Figure 14.

The use of DL techniques makes biparametric analysis faster: with the classical technique (LEs) we need approximately 25 hours to obtain the biparametric plane of Figure 11A (of $1000 \times 1000$ points) and with DL (in the same machine) around 30 minutes are necessary to generate it (see Table VI for more details). Note that LEs usually take a long time to converge to their correct value.

In Figure 11 we have the study of regular and chaotic behaviour in the biparametric plane $(r,b)$ with $\sigma = 10$ of the Lorenz system. In panel A, classification is performed according to the first approximated LE (white for chaotic and black for regular). On the left side of panels B, C and D chaos detection with the DL architectures (MLP, CNN and LSTM, respectively) is depicted (same code of colours of panel A). On the right side of such panels we have the errors committed by the networks (we have compared with the biparametric plot obtained with LEs in panel A). In particular, chaotic samples wrongly classified (false regular, $F_R$) are in red, misclassified regular samples (false chaotic, $F_C$) are in blue, and samples detected correctly (true regular, $T_R$, and true chaotic, $T_C$) are in white. Moreover, for each technique, we have indicated the accuracy (percentage of correct detections) which is always greater than 94% and, as expected, it is bigger for the CNN and the LSTM networks.

As it can be seen in error plots, most of the incorrect detections correspond to the right boundary between chaotic and regular regimes (in the case of the MLP, a lot of noise can also be seen for $r$ small where there are $F_C$ detections). The $F_R$ detections stand out in this right zone for the MLP and the LSTM cases, and $F_C$ results highlight for the CNN, being the LSTM the network with the lower number of errors. This right boundary have to be studied carefully with whatever technique used for chaos detection as a dynamical behaviour known as transient chaos[8] occurs. In panels A1, B1, C1 and D1 a zoom of the green framed region that includes part of this boundary is depicted for each technique.

p2_def_nueva.pdf

Figure 12. In panel A, chaos detection for the Lorenz system in the biparametric plane $(r,\sigma)$ with $b = 8/3$. From left to right: results obtained with LEs, detections with the trained LSTM network, and errors committed by this architecture (same colour code than in Figure 11 is used to represent regular and chaotic behaviour, and false regular and false chaotic detections). The percentage corresponds to the accuracy of the LSTM for the chaos detection task. The orange horizontal line shows the samples present in the raw datasets used to create the training set. In panels A1 and A2, a zoom of the green framed region of the biparametric plane has been represented using the classical and DL techniques. The yellow point corresponds to an orbit in Figure 14.

The three DL architectures are able to study the behaviour of the Lorenz system in the biparametric plane $(r,b)$ where the three $r$-parametric lines used during training process are included (see orange and purple lines in Figure 11A). Now, we show that DL is also able to reconstruct other biparametric planes of the parameter space. In particular, we compute the biparametric plane $(r,\sigma)$ for $b = 8/3$, $r \in [1,350]$ and $\sigma \in [0,60]$; the $(\sigma,b)$ region for $r = 28$, $\sigma \in [0,40]$ and $b \in [0,4]$; the $(\sigma,b)$ plane for $r = 500$, $\sigma \in [0,800]$ and $b \in [0,100]$. To perform this task we use the LSTM as it gives the best results in the plane $(r,b)$ (see Figure 11).

In panel A of Figure 12 we have, from left to right, the biparametric plane $(r,\sigma)$ obtained with the approximated LEs, the results of the trained LSTM, and the errors committed by this network. Same code of colours as in Figure 11 is used. The boundaries between both dynamical regimes are well-defined by the LSTM, only the top boundary is noisy (see panel A2 for a zoom of the green framed region where it is included, and panel A1 for the corresponding results of LEs) and $F_C$

detections stand out. The percentage of correct detections (accuracy) is greater than 97%. Notice that in this case, where the chaotic detection is almost perfect, only the samples in the orange segment belong to the raw datasets used to create the training set. In panel A of Figure 13, we have the study of the biparametric plane $(\sigma, b)$ with $r = 28$. Again, we can see that this network is able to define correctly the boundary between both dynamical regimes. It highlights the bottom right corner, a regular zone that has been defined as chaotic by the LSTM. In panel A2 of Figure 13, that corresponds to the green framed region, it is shown that the trained LSTM performs the task with high precision (panel A1 corresponds to LEs). In fact, the percentage of correct detections in the whole plane is almost 98%. In this case, only three points are in the raw datasets used to create train and validation sets, which shows the high generalization ability of the trained LSTM. In panel B of Figure 13 the biparametric analysis $(\sigma, b)$ with $r = 500$ is depicted. As in the other panels, the trained LSTM network distinguishes correctly both dynamical regimes, as it can be seen in panel B2 which is a zoom of the green framed region (see panel B1 for detection performed with LEs). The accuracy of this study is greater than 98%. Most of the errors of the network correspond to small values of $b$. Notice that no point of this parametric region has been used to train (or validate) the network. With the results of Figures 12 and 13 we can conclude that DL allows us to study regions of the parameter space where training and validation points are almost not present (Figure 12A and Figure 13A) or are not present at all (Figure 13B).

Analogously to what we did in the case of the Logistic map, we analyse in Figure 14 some correct and incorrect classifications of DL networks (see yellow dots in Figures 11, 12 and 13). In panel I of Figure 14 we show a periodic orbit (regular behaviour) that DL networks have not been able to classify adequately. The dynamical explanation is due to the nature of this orbit, it is a periodic orbit with a large number of space-expanding loops and a large period. The sample in panel II is chaotic and the networks have also failed to classify it properly. Now the explanation is due to the kind of chaos of this orbit. The orbit changes slowly in space, being the different loops very similar each other (see the 3D plot of the DL data). All the DL techniques fail to identify the kind of behaviour of orbits I and II, but there are clear reasons for the fail, and in fact any short time methodology would fail on these orbits. Besides, both cases are in the limit between the regular and chaotic dynamics with a lot of bifurcations located in that area. In contrast, all networks have managed to properly identify the type of behaviour of the orbits in panels III and V, regular and chaotic, respectively. As an intermediate situation, we show the regular orbits IV and VI for which only LSTM in the first case, and CNN and LSTM in the second have been able to give a correct classification. The complicated points have been selected on the basis of the previous analyses performed on the biparametric planes of Figures 11, 12 and 13.

As a final and challenging task, we study a 3D parametric space region of the Lorenz system (allowing the three parameters to change) with the LSTM trained and validated using only three $r$-parametric segments (see orange and purple lines in Figure 11). Note that with the classical technique of LEs this kind of analysis would be highly computationally demanding. Taking into account the time that we need to compute a 2D diagram with LEs, the expected time for the 3D study in this subsection (with $250 \times 250 \times 250$ points) with this classical technique would be of 16 days approximately; with DL we need around 12 hours (see Table VI). We also remark that, up to the knowledge of the authors, there is no dense 3D analysis in the literature for the Lorenz system. In Refs. 1 and 2 the authors studied several 2D planes to reconstruct a 3D figure, but it is not a complete dense 3D plot as in the present work. Again, we remark that we use just a very few lines of data to train and generate a network able to completely perform a 3D parametric study. Therefore, these techniques open a window for dense 3D parametric studies in a reasonable time.

The 3D study is performed for $\sigma \in [0,800]$, $r \in [1,500]$ and $b \in [0,100]$, taking 250 values for each parameter. In panel A of Figure 15 we have used the detections of the trained LSTM network to illustrate a 3D parametric study of the Lorenz system. In particular, for a better representation we have drawn the boundary between both dynamical regimes of the obtained dense 3D DL figure, and the chaotic results for $r = 500$. In panels B and C, we have several 2D planes extracted from the 3D study (in shaded black we have the boundary between both dynamical regimes). For completion, in Figure 16 we show two different views of the boundary between regular and chaotic regimes detected by the DL network.

Notice that the trained LSTM detects perfectly the boundary (represented in detail in Figure 16) between both dynamical regimes. In Figure 16 we have used both methodologies, the Lyapunov exponents and the LSTM network, and the resulting figures are indistinguishable. The dense 3D computation using the Lyapunov exponents has detected 1852204 set of parameters of chaotic dynamics, whereas the LSTM network 1855862, being the intersection 1827514 points. The number of false positives is 28348 and the number of false negatives is 24690. Moreover, from the 2D planes in Figure 15 it can be inferred that DL is able to detect the changes between regular and chaotic behaviours that are present for small values of parameter $b$. We conclude that with an LSTM trained (and validated) in a small region of the parameter space ($\sigma = 10$, $r \in (0,300]$, $b \in \{2, 2.4, 8/3\}$), an accurate dense 3D analysis can be performed.

As a complement, we include in the Integral Multimedia (Multimedia view) of the article a video where we show the dense 3D parametric study that has allowed analyses in Figure 15, and the chaotic boundary represented in Figure 16.

| | **MLP** | **CNN** | **LSTM** |
|---|---|---|---|
| **Creation of each raw dataset** (CPU with parallel computing) | 419 | 419 | 419 |
| **Data selection of train, validation and test sets** (CPU) | 882.586 | 882.586 | 882.586 |
| **Normalization of train, validation and test sets** (CPU) | 15.833 | 15.833 | 15.833 |
| **Training** (CUDA with PyTorch) | 2171.203 | 2154.514 | 8601.143 |
| **Test** (CUDA with PyTorch) | 0.158 | 0.208 | 0.461 |
| **One-parameter line. Create data** (CPU with parallel computing) | 1.571 | 1.571 | 1.571 |
| **One-parameter line. Prepare data** (CPU) | 8.348 | 9.427 | 9.474 |
| ♣ **One-parameter line. Detection** (CUDA with PyTorch) | 0.012 | 0.016 | 0.072 |
| **Biparametric plane. Create data** (CPU with parallel computing) | 307.940 | 307.940 | 307.940 |
| **Biparametric plane. Prepare data** (CPU) | 1418.089 | 1636.286 | 1623.851 |
| ♣ **Biparametric plane. Detection** (CUDA with PyTorch) | 3.729 | 37.434 | 17.630 |
| **Triparametric analysis. Create data** (CPU with parallel computing) | – | – | 14156.416 |
| **Triparametric analysis. Prepare data** (CPU) | – | – | 28929.904 |
| ♣ **Triparametric analysis. Detection** (CUDA with PyTorch) | – | – | 1139.313 |
| **One-parameter line. Classical technique LEs** (CPU with parallel computing) | 419 | | |
| **Biparametric plane. Classical technique LEs** (CPU with parallel computing) | $9 \cdot 10^4$ | | |
| **Triparametric analysis. Classical technique LEs** (CPU with parallel computing) | $1.40213 \cdot 10^6$ | | |

Table VI. Approximated time (in seconds) needed for the chaos detection task in the Lorenz system test problem (from data creation to 1D, 2D and 3D parametric analysis for each DL network). The pure chaos detection DL process, once the data is ready, is just the `Detection(CUDA with PyTorch)` time, pointed with a club symbol. The last three rows correspond to the time used by the classical technique of Lyapunov Exponents.

To give an idea of the time savings of using DL networks, we show in Table VI the times (in seconds) needed during the whole study of the Lorenz system test problem. The first three rows correspond to the time used to create train, test and validation datasets, which includes creation of raw datasets, data selection, and normalization. Fourth and fifth rows correspond to training and test processes. To obtain the one-parameter line of Figure 8, which contains 6000 different values of $r$, we indicate the time needed to create the data, to prepare it as network input, and to be classified by the ANN (sixth, seventh and eighth rows). The whole one-parameter study takes less than 12 seconds for all the networks. Notice, that the pure chaos detection DL process (if data is given) corresponds to row 8 (club symbol) and is less than one tenth of a second. With classical techniques (third-to-last row) needed time is around 7 minutes. In the case of biparametric planes, as these of Figures 11, 12 and 13 with $1000 \times 1000$ points, whole process (rows 9 to 11) takes approximately 30 minutes for all the networks, with a pure chaos detection (row 11 with club symbol) of less than 40 seconds. Time used by classical techniques is given in second-to-last row and it is of 25 hours. Finally, the triparametric study of Figures 15 and 16 with $250 \times 250 \times 250$ points takes around 12 hours for the LSTM network (see rows 12 to 14). In this case, the pure chaos detection process (row 14 with a club symbol) is less than 20 minutes. In the last row of the table we have the time needed to perform the triparametric analysis with classical techniques. Notice that this analysis with DL takes approximately 12 hours, but we would need more than 16 days if classical techniques were used.

## V. DISCUSSION AND CONCLUSIONS

In this paper, three well-known Deep Learning networks (MLP, CNN and LSTM) have been built and trained to carry out the chaos detection task in two test problems: the Logistic map (discrete dynamical system) and the Lorenz system (continuous dynamical system). Usually, time-consuming computational techniques, such as Lyapunov Exponents, are used to detect chaos.

All the trained networks have given good results, achieving all of them an accuracy greater than 90%. Moreover, in all the cases the accuracy of detection of both dynamical regimes (regular and chaotic) is balanced. In the case of the Logistic map test problem, the accuracy for the three networks is very similar. However, in the case of the Lorenz system, the CNN and the LSTM give better results than the MLP. This makes sense since CNNs and LSTM networks take into account spatial and temporal information, respectively.

For the Lorenz system, we have used the trained networks to study the behaviour of an $r$-parametric line and several biparametric planes from different regions of the parameter space. In addition, a dense triparametric plot of the Lorenz system has also been obtained using a trained

LSTM network. Up to the knowledge of the authors, this had not been achieved previously with classical or DL techniques. We highlight that the training process used just a few lines of one-parameter data to create a network capable of studying biparametric and even complete triparametric spaces. This is a remarkable result that shows us the power of DL techniques in dynamical systems studies.

In this article we focus on using short time series as data, but longer sequences will give us more complete information, particularly in difficult areas where the LE value is close to zero. To achieve better results, in future research we will explore the development of improved architectures that are better suited to handle longer sequences or carefully select more challenging training examples to reduce error rates. However, it is important to balance this against the computational cost, as our current focus has been to keep the computational cost low compared to traditional calculation of the LEs.

We conclude that Deep Learning can be used to analyse the behaviour (regular and chaotic) of a dynamical system. Our results show that even dense 3D parametric studies can be carried out in a very reasonable time using data from just a small portion of the global phase space. However, a deeper study would be necessary to know how far we can go using these techniques in this and others dynamical systems tasks.

## ACKNOWLEDGMENTS

RB, AL, AMC, CMC and SS have been supported by the Spanish Research projects PGC2018-096026-B-I00 and PID2021-122961NB-I00. RB and SS have been supported by the European Regional Development Fund and Diputación General de Aragón (E24-20R and LMP94-21). AL has been supported by Diputación General de Aragón project E22-20R. AMC has been supported by Diputación General de Aragón with a PhD student grant. CMC has been supported by Ministerio de Universidades de España with an FPU grant (FPU20/04039). AM and AO have been supported by Diputación General de Aragón (Grant Group T36-20R: Vivolab). RV has been supported by the Spanish Research project PID2021-122961NB-I00.

## DATA AVAILABILITY

The training, validation and test datasets (for both test problems: Logistic map and Lorenz system), as well as the original data used to obtain them, can be found in the free Mendeley Data repository in the link:

`https://doi.org/10.17632/k4x675k5dm.2`

## REFERENCES

[1] R. Barrio and S. Serrano. A three-parametric study of the Lorenz model. *Physica D*, 229(1):43–51, 2007.

[2] R. Barrio and S. Serrano. Bounds for the chaotic region in the Lorenz model. *Physica D*, 238(16):1615–1624, 2009.

[3] N. Boullé, V. Dallas, Y. Nakatsukasa, and D. Samaddar. Classification of chaotic time series with Deep Learning. *Physica D*, 403:132261, 2020.

[4] J. Bragard, H. Pleiner, O. J. Suarez, P. Vargas, J. A. C. Gallas, and D. Laroze. Chaotic dynamics of a magnetic nanoparticle. *Physical Review E*, 84(3):037202, 2011.

[5] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković. Geometric Deep Learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

[6] A. Celletti, C. Gales, V. Rodriguez-Fernandez, and M. Vasile. Classification of regular and chaotic motions in Hamiltonian systems with Deep Learning. *Scientific Reports*, 12(1):1–12, 2022.

[7] R. Chandra, S. Goyal, and R. Gupta. Evaluation of Deep Learning models for multi-step ahead time series prediction. *IEEE Access*, 9:83105–83123, 2021.

[8] E. J. Doedel, B. Krauskopf, and H. M. Osinga. Global invariant manifolds in the transition to preturbulence in the Lorenz system. *Indagationes Mathematicae*, 22(3-4):222–240, 2011.

[9] A. Géron. *Hands-On Machine Learning with Scikit-Learn* & *TensorFlow*. O'Reilly Media, Inc., Sebastopol, 2019.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] C. F. Higham and D. J. Higham. Deep Learning: An introduction for applied mathematicians.

*SIAM review*, 61(4):860–891, 2019.

[12] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural computation*, 9(8):1735–1780, 1997.

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324, 1998.

[14] W. S. Lee and S. Flach. Deep Learning of chaos classification. *Machine Learning: Science and Technology*, 1(4):045019, 2020.

[15] E. N. Lorenz. Deterministic nonperiodic flow. *Journal of atmospheric sciences*, 20(2):130–141, 1963.

[16] R. M. May. Simple mathematical models with very complicated dynamics. *Nature*, 261(5560):459–467, 1976.

[17] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T Killeen, L. Zeming, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An imperative style, high-performance Deep Learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[18] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4780–4789, 2019.

[19] P. R. Vlachas, J. Pathak, B. R. Hunt, T. P. Sapsis, M. Girvan, E. Ott, and P. Koumoutsakos. Backpropagation algorithms and Reservoir Computing in Recurrent Neural Networks for the forecasting of complex spatiotemporal dynamics. *Neural Networks*, 126:191–217, 2020.

[20] A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano. Determining Lyapunov exponents from a time series. *Physica D*, 16(3):285–317, 1985.

p3_def_nueva.pdf

Figure 13. In panel A, chaos detection for the Lorenz system in the biparametric plane $(\sigma,b)$ with $r=28$. In panel B, chaos detection for the Lorenz system in the biparametric plane $(\sigma,b)$ with $r=500$. From left to right: results obtained with LEs, detections with the trained LSTM network, and errors committed by this architecture (same colour code than in Figure 11 is used to represent regular and chaotic behaviour, and false regular and false chaotic detections). The percentage given in each panel corresponds to the accuracy of the LSTM for such biparametric plane. The orange and purple points show the samples present in the raw datasets used to create the training and validation sets. In panels A1, A2, B1 and B2, a zoom of the green framed region of the biparametric plane has been represented using the classical and DL techniques. The yellow point corresponds to an orbit in Figure 14.

Orbitas2a.pdf

Figure 14. Some orbits (in grey) of the Lorenz system with its true dynamical behaviour, the corresponding sample (in red) used by the networks and the correctness or incorrectness of the classification made by them. In all the cases $(x_0,y_0,x_0)=(1,1,1)$. The values of the parameters are (I) $(\sigma,r,b)=(10,267.84464,2)$, (II) $(\sigma,r,b)=(10,246.3910675,2.2)$, (III) $(\sigma,r,b)=(10,220,2.71)$, (IV) $(\sigma,r,b)=(10,208.3006,2.8090)$, (V) $(\sigma,r,b)=(45.3,160,8/3)$ and (VI) $(\sigma,r,b)=(31,28,0.8)$. See yellow points in Figures 11, 12 and 13.

3D_planes.pdf

Figure 15. Representations obtained from the dense triparametric analysis of the Lorenz system performed with the LSTM network. See Integral Multimedia (Multimedia view) for the dense study. In panel A, general view of the upper boundary between both dynamical regimes, and chaotic region for $r=500$. In panels B and C, some 2D vertical and horizontal planes in the triparametric space with the chaotic detections in colour and the external boundary between both dynamical regimes in shaded black. The colors used have been simply selected for ease of viewing.

3Dsurface.pdf

Figure 16. Detail of the boundary between regular and chaotic regions from a dense 3D parametric analysis of the Lorenz system performed with the Lyapunov exponents and with the LSTM network. See Integral Multimedia (Multimedia view) for an integral view and the dense analysis.